I hereby certify that this correspondence is being mailed by first class mail with sufficient postage to Commissioner for Patents
P.O. Box 1450, Alexandria, VA 22313-1450,
on: September 30, 2005.

Robert Moll, Registration No. 33,741

U.S. PATENT AND TRADEMARK OFFICE

In re Application of: Michael Lee Workman et. al.
Title: Systems and Methods of Multiple Access Paths to Single Ported Storage Devices
Application No. 10/677,560
Filing Date: October 1, 2003

Examiner: Ilwoo Park
Art Unit: 2182
Attorney Docket No. Pillar 716

DECLARATION OF PAUL THOMAS PETERSEN

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

I, Paul Thomas Petersen, declare as follows:

I am an engineer at Pillar Data Systems, Inc. since March 2002.

I am co-inventor of claims 1 and 23 of the above-identified patent application.

I am also co-inventor of US Application No. 10/264,603, entitled, Systems and Methods of Multiple Access Paths to Single Ported Storage Devices, filed on October 3, 2002, which describes a Serial ATA coupling circuit without a microcontroller.

Prior to March 20, 2003, during a design review of the Serial ATA coupling circuit without a microcontroller, Douglas John Fox and I conceived of adding a microcontroller to control the coupling circuit switches of the Serial ATA coupling circuit.

Also prior to March 20, 2003, I finished a block diagram for the Serial ATA coupling circuit with a microcontroller. A copy is attached as Exhibit A.

Also prior to March 20, 2003, I reviewed a document entitled Storage Enclosure Mudflap Requirements describing requirements for a Serial ATA coupling circuit with a microcontroller. A copy is attached as Exhibit B.

On April 7, 2003, I submitted an engineering change order (ECO #E1068) to Pillar document control. On April 10, 2003, the ECO was approved. This enabled Pillar to engage an outside company to manufacture the printed circuit board assembly implementing a Serial ATA coupling circuit with a microcontroller. A copy is attached as Exhibit C.

On April 22, 2003, I received the first printed circuit board assemblies back from the manufacturer. I loaded firmware that I obtained from Douglas John Fox into the flash memory of the microcontroller and using an oscilloscope verified the printed circuit assemblies successfully implemented the Serial ATA coupling circuit as recited in claim 1 of the present application.

Each of the dates deleted from Exhibits A and B is prior to March 20, 2003.

I am warned that willful false statements and the like are punishable by fine or imprisonment, or both (18 U.S.C. 1001) and may jeopardize the validity of the application or any patent issuing thereon. All statements made of the declarant's own knowledge are true and that all statements made on information and belief are believed to be true.

Paul Thomas Petersen

Principal Engineer

Pillar Data Systems, Inc.

San Jose, California

I hereby certify that this correspondence is being mailed by first class mail with sufficient postage to Commissioner for Patents
P.O. Box 1450, Alexandria, VA 22313-1450,
on: September 30, 2005.

Robert Moll, Registration No. 33,741

U.S. PATENT AND TRADEMARK OFFICE

In re Application of: Michael Lee Workman et. al.
Title: Systems and Methods of Multiple Access Paths to Single Ported Storage Devices
Application No. 10/677,560
Filing Date: October 1, 2003

Examiner: Ilwoo Park
Art Unit: 2182
Attorney Docket No. Pillar 716

DECLARATION OF DOUGLAS JOHN FOX

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

I, Douglas John Fox, declare as follows:

I am a principal firmware engineer at Pillar Data Systems, Inc. since January 2003.

I am co-inventor of claims 1 and 23 of the above-identified patent application.

Prior to March 20, 2003, during a design review of the Serial ATA coupling circuit without a microcontroller, Paul Thomas Petersen and I conceived of adding a microcontroller to a Serial ATA coupling circuit to control the coupling circuit switches.

During March 2003, I reviewed a block diagram prepared by Paul Thomas Petersen showing a Serial ATA coupling circuit with a microcontroller. A copy is attached as Exhibit A.

During March 2003, I reviewed a document entitled Storage Enclosure Mudflap Requirements prepared by Paul Thomas Petersen describing requirements for a Serial ATA coupling circuit with a microcontroller. A copy is attached as Exhibit B.

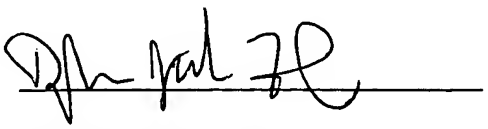
From March to April 2003, I spent time with Paul Thomas Petersen discussing the hardware and firmware requirements to implement a Serial ATA coupling circuit with a microcontroller. I selected the microcontroller, reviewed host driver code required to communicate with the Serial ATA coupling circuit and defined the firmware architecture of the microcontroller.

In April 2003, I began writing the firmware to control the Serial ATA coupling circuit with microcontroller resulting in a set of files. A copy of a directory list of those files is attached as Exhibit E.

By May 2003, I had completed and tested the firmware to control the Serial ATA coupling circuit. It worked for its intended purpose that is the microcontroller was programmed to control the coupling circuit switches of the Serial ATA coupling circuit. A copy of the firmware is attached as Exhibit F.

Each of the dates deleted from Exhibits A and B is prior to March 20, 2003.

I am warned that willful false statements and the like are punishable by fine or imprisonment, or both (18 U.S.C. 1001) and may jeopardize the validity of the application or any patent issuing thereon. All statements made of the declarant's own knowledge are true and that all statements made on information and belief are believed to be true.



Douglas John Fox
Principal Firmware Engineer
Pillar Data Systems, Inc.
San Jose, California

Page #	Description
1	COVER PAGE
2	MUX & CONNECTORS
3	DRIVE ENABLE LOGIC
4	HOT SWAP CONTROLLER
5	POWER & MISC
6	NOTES

ECO #	Description

Block Diagram
Active Mudflap

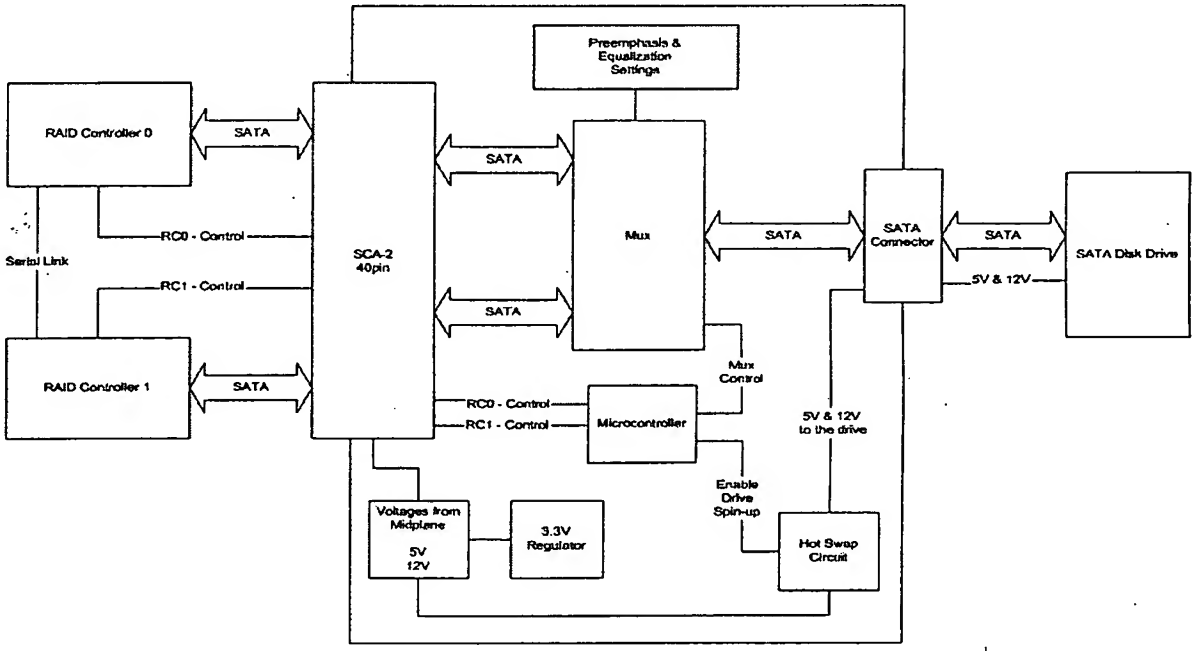


EXHIBIT A

Note: The top (or component) side of the board is the side where the SCA2 40pin connector is located.

bom-Rev 1.0

Drawn By: Paul Petersen

pillar DATA SYSTEMS

1371 McCarthy Boulevard, Milpitas, CA 95035

Title: Mudflap w/ Maxim Mux

Size: B | Document #: 1700-00350-01SCH | Rev: 1.0

Date: April 04, 2003 | Sheet 1 of 6

Storage Enclosure Mudflap Requirements

Document Owner:	Paul Petersen
Date:	March 25, 2003
Version:	0.3

1 Table of Contents

2 Revision History

Version	Date	Description
0.1		Initial Revision
0.2		Added command protocol. Changed power on circuit to power on/off circuit. Added support & drawings. Converted white on black drawings to black on white.
0.3	3/25/2003	Imported Eurologic's PCB outline drawing. Deleted the PDS stiffener drawings as the latest improved stiffener design from Eurologic looks to meet our requirements. Changed the PCB thickness to 0.055".

3 Supporting Documentation

- Serial ATA: High Speed Serialized Attachment, Rev. 1.0, August 29, 2001

4 Mudflap Requirements

The mudflap is a small pcba that interposes between the disk drive and midplane. It is mounted near the disk drive and is attached to the disk drive carrier. The main function of the mudflap is to direct Serial ATA (SATA) traffic between the disk drive and the selected RAID Controller.

Twelve of these mudflaps are required per Storage Enclosure. A 13th mudflap is of a different form-factor and has a slightly different pinout and functionality. The 13th disk drive mudflap requirements are not represented in this document.

4.1 Feature Summary

Mudflaps provide the following functions:

- 2 to 1 multiplexing of the SATA signals between both RAID Controllers and the disk drive
- Power on/power off control of the disk drive
- Suppression of power transients during hot insertion and hot removal of a mudflap

4.2 SATA Multiplexing

The mudflap must allow the RAID Controllers to select the path the SATA data takes to get/from the disk drive. There shall be no priority. Who ever asked for the data path last gets it.

The power-on reset state of the mux control circuit must guarantee that SATA data is routed to RAID Controller 1.

4.3 Disk Drive Power Control

The mudflap must not allow the disk drive to spin up when power is applied to the enclosure. Currently, SATA drives spin up when power is applied to them so the mudflap must have circuitry to interrupt the flow of current to the disk drive. Only after receipt of a proper command, must the mudflap allow the disk drive to spin up.

4.4 Transient Suppression

Tthe mudflap must have circuitry to prevent the midplane voltages (5V & 12V) from overshooting when a drive carrier is removed while the storage enclosure is powered and the disk drive is still spinning.

4.5 Mudflap Control

Each RAID Controller drives a separate mudflap control signal that is open drain and must terminate into a Schmitt trigger buffer located on the mudflap. There shall also be a 4.7k Ohm pullup to 3.3V on this signal. The pullup shall be located on the mudflap.

Commands are issued to the mudflap using a one-wire serialized packet protocol. See section 5 for the allowed commands and the specified command format.

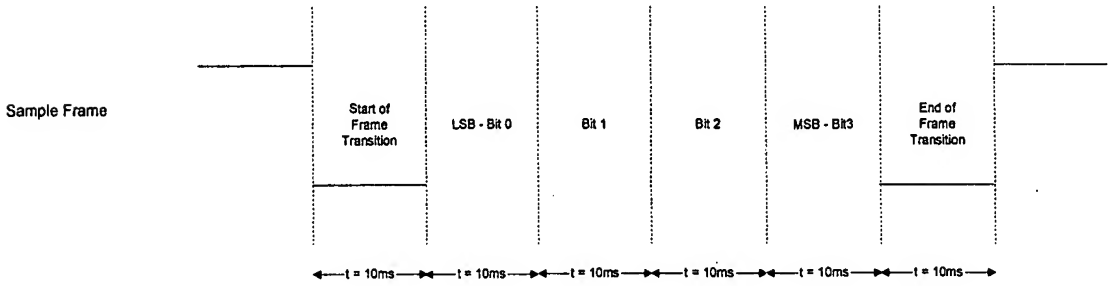
The mudflap controller shall respond to commands in the order received. If a command is not recognizable, it shall be ignored. It is not permissible to ignore a properly formatted command.

The mudflap controller shall have a power-on reset circuit and a watchdog timer that resets the mudflap controller if the watchdog detects a failure.

5 Command Encoding

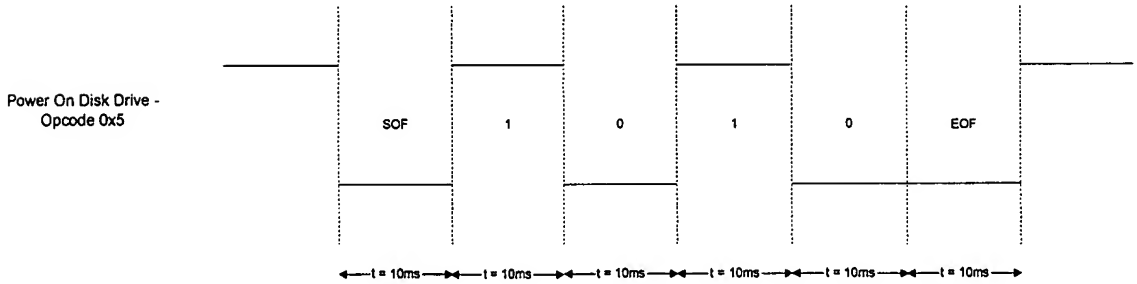
A command frame consists of a start of frame transition, four data bits and an end of frame transition. The LSB is the first data bit transmitted. The bit cell time is 10ms. The microcontroller shall oversample the frame by at least 8x.

The frame structure is shown below.

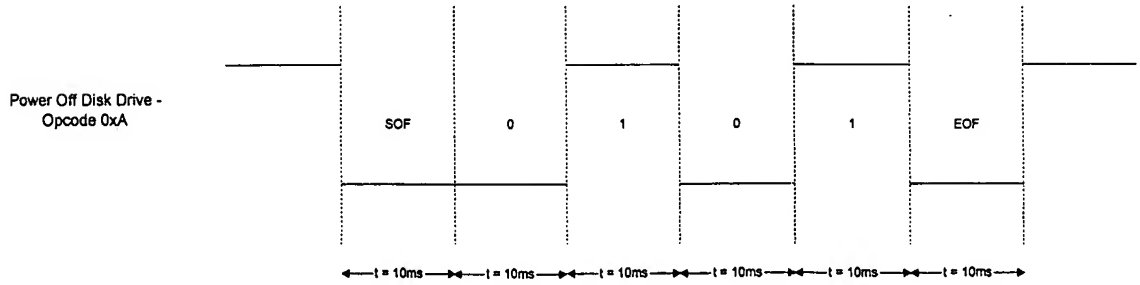


The four allowable command frames are shown in the following sections.

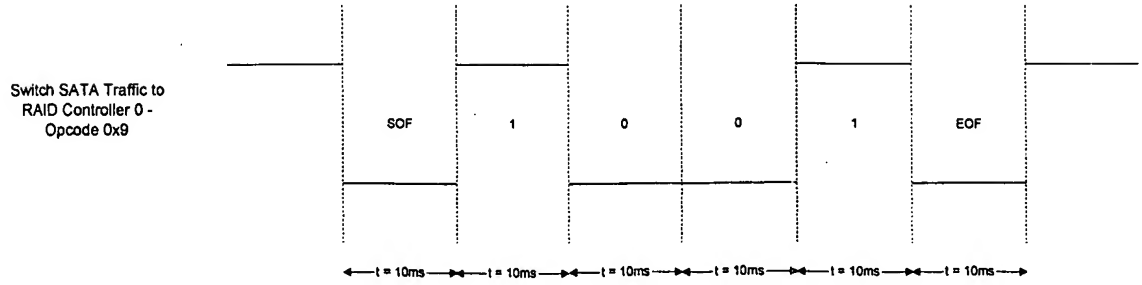
5.1 Power On Disk Drive Frame



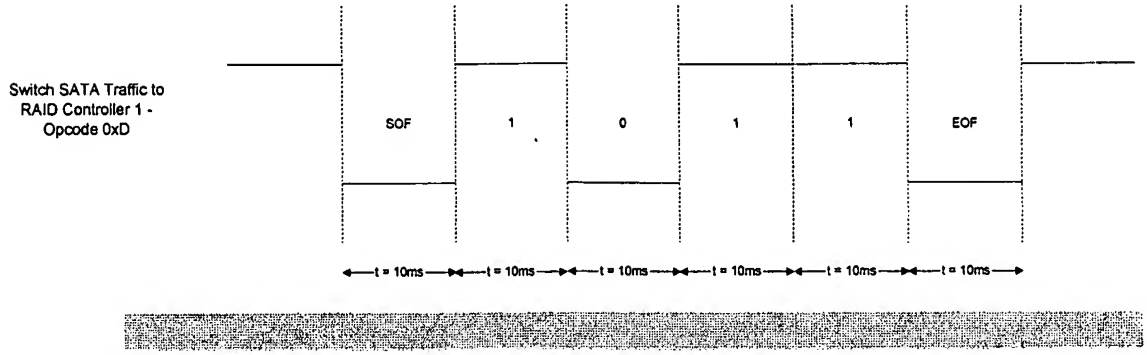
5.2 Power Off Disk Drive Frame



5.3 Switch SATA TO RAID Controller 0 Frame



5.4 Switch SATA to RAID Controller 1 Frame



6 Mudflap Connector Pinouts

6.1 Midplane Connector Pinout & Signal Definitions

The mudflap connects to the midplane using a SCA-2 40 position vertical plug (AMP 84488-3).

Pin(s)	Signal Name & Description
1	N/C
2	12V
3	12V
4	12V
5	N/C
6	Disk Drive Present Signal – Must be connected to ground on the mudflap.
7 - 16	N/C
17	Preemphasis/Equalization Control – RAID Controller 0 – This signal is slot specific and may be open or connected to ground through a 100Ω resistor on the midplane.
18	Preemphasis/Equalization Control – RAID Controller 1 – This signal is slot specific and may be open or connected to ground through a 100Ω resistor on the midplane.
19	5V
20	5V
21	12V
22	Ground
23	Ground
24	RAID Controller 0 SATA+ to disk drive– This is the '+' side of the SATA differential pair that originates at RAID Controller 0, passes through the mux and is received at the drive.
25	RAID Controller 0 SATA- to disk drive– This is the '-' side of the SATA differential pair that originates at RAID Controller 0, passes through the mux and is received at the drive.
26	Ground
27	RAID Controller 1 SATA+ to disk drive– This is the '+' side of the SATA differential pair that originates at RAID Controller 1, passes through the mux and is received at the drive.

28	RAID Controller 1 SATA- to disk drive– This is the '-' side of the SATA differential pair that originates at RAID Controller 1, passes through the mux and is received at the drive.
29	Ground
30	RAID Controller 0 SATA+ from disk drive– This is the '+' side of the SATA differential pair that originates at the drive, passes through the mux and is received at RAID Controller 0.
31	RAID Controller 0 SATA- from disk drive– This is the '-' side of the SATA differential pair that originates at the drive, passes through the mux and is received at RAID Controller 0.
32	Ground
33	RAID Controller 1 SATA+ from disk drive– This is the '+' side of the SATA differential pair that originates at the drive, passes through the mux and is received at RAID Controller 1.
34	RAID Controller 1 SATA- from disk drive – This is the '-' side of the SATA differential pair that originates at the drive, passes through the mux and is received at RAID Controller 1.
35	Ground
36	N/C
37	N/C
38	Mux Control – RAID Controller 1 –When the "mux switch to RC 1" command is sent to the mudflap, the SATA data is routed between the disk drive and RAID Controller 1.
39	Mux Control – RAID Controller 0 – When the "mux switch to RC" command is sent to the mudflap, the SATA data is routed between the disk drive and RAID Controller 0.
40	5V

6.2 SATA Disk Drive Connector Pinout & Signal Definitions

The mudflap connects to the disk drive using an extended height SATA connector (Molex 87678-0001).

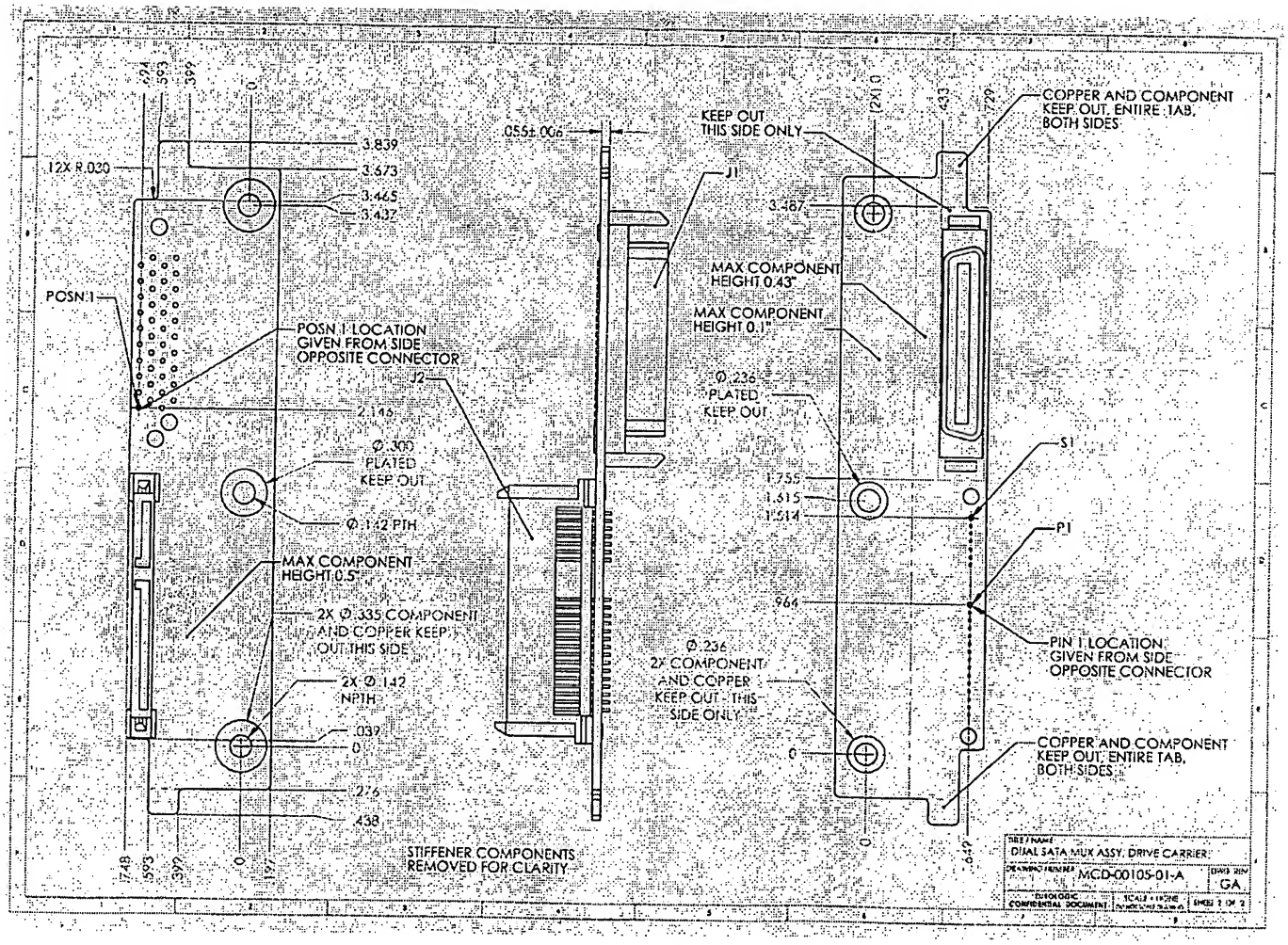
Pin(s)	Signal Name & Description
S1	Ground
S2	SATA Rx+ – This is the '+' side of the SATA differential pair that originates at the selected RAID Controller, passes through the SATA multiplexer and is received at the disk drive.
S3	SATA Rx- – This is the '-' side of the SATA differential pair that originates at the selected RAID Controller, passes through the SATA multiplexer and is received at the disk drive.
S4	Ground
S5	SATA Tx- – This is the '-' side of the SATA differential pair that originates at the disk drive, is routed through the mux and is received at the selected RAID Controller.
S6	SATA Tx+ – This is the '+' side of the SATA differential pair that originates at the disk drive, is routed through the SATA mux and is received at the selected RAID Controller.
S7	Ground
P1 – P3	N/C
P4	Ground
P5	Ground
P6	Ground
P7	5V
P8	5V
P9	5V
P10	Ground
P11	N/C (Reserved)
P12	Ground
P13 – P15	12V

7 PCB Mechanical Dimensions

7.1 PCB Thickness

The mudflap PCB is to have a finished thickness of 0.055" (± 0.006").

7.2 PCB Mechanical Outline Drawing



pillar™ DATA SYSTEMS	**ENGINEERING CHANGE ORDER COVER SHEET**	**ECO:** E1068 **LIFECYLE:** Pre-Production
ORIGINATOR: Brad Burkhart **DATE:** April 9, 2003	**PRIORITY:** Normal **PROJECT:** Brix	**RELEASE:** 4/10/03 **EFFECTIVE:** 4/10/03

REASON FOR CHANGE: New Release.

DESCRIPTION OF CHANGE: Initial Engineering release of the Active Mudflap w/Maxim Mux Board package (1700-00350-01xxx) Rev 1.0 to Doc Control.

The old PCBA, Active Mudflap (1700-00350-00xxx) and associated files are being obsoleted in this ECO.

APPROVALS:

Originator:	Date:	*Engineering:	Date:	*Materials:	Date:
Purchasing:	Date:	*Manufacturing Eng:	Date:	Assembly Test:	Date:
Marketing:	Date:	Cost Accounting:	Date:	*Doc Control:	Date:

* = Required signatures for Pre-Production ECO's.

CCB COMMENTS: Electronically signed-off in Oracle.

EXHIBIT C

pillar™ DATA SYSTEMS

ENGINEERING CHANGE ORDER AFFECTED ITEMS SHEET

ECO: E1068

Part Number	Old Rev	New Rev	Description	Material Dispositions						
				On Order	Stk	WIP	FGI	Field	Repair	
1700-00350-01ASM	NA	1.0	PCBA Dwg, Active Mudflap w/Maxim Mux	N	N	N	NA	NA	NA	
1700-00350-01PCA	NA	1.0	PCBA, Active Mudflap w/Maxim Mux	N	N	N	NA	NA	NA	
1700-00350-01FAB	NA	1.0	PCB Fab, Active Mudflap w/Maxim Mux	N	N	N	NA	NA	NA	
1700-00350-01GBR	NA	1.0	PCB Gerber Files, Active Mudflap w/Maxim Mux	N	N	N	NA	NA	NA	
1700-00350-01SCH	NA	1.0	PCBA Sch, Active Mudflap w/Maxim Mux	N	N	N	NA	NA	NA	
1700-00350-00ASM	1.0	OBS	PCBA Dwg,Active Mudflap w/Maxim Mux *(Obsolete-replaced by 1700-00350-01ASM)*	S	S	S	NA	NA	NA	
1700-00350-00PCA	1.0	OBS	PCBA,Active Mudflap w/Maxim Mux *(Obsolete-replaced by 1700-00350-01PCA)*	S	S	S	NA	NA	NA	
1700-00350-00FAB	1.0	OBS	PCB Fab,Active Mudflap w/Maxim Mux *(Obsolete-replaced by 1700-00350-01FAB)*	S	S	S	NA	NA	NA	
1700-00350-00GBR	1.0	OBS	PCB Gerber file,Active Mudflap w/Maxim Mux *(Obsolete-replaced by 1700-00350-01GBR)*	S	S	S	NA	NA	NA	
1700-00350-00SCH	1.0	OBS	PCBA Shem,Active Mudflap w/Maxim Mux *(Obsolete-replaced by 1700-00350-01SCH)*	S	S	S	NA	NA	NA	
2100-01583-00	NA	A	CONN,HDR,6 POS,1.25MM,PITCH RIGHT ANGLE	N	N	N	NA	NA	NA	
3130-02614-00	NA	A	IC,NC7WB66,2 BIT BUS SWITCH,US8	N	N	N	NA	NA	NA	
3130-02615-00	NA	A	IC,ATTINY12L,AVR MICROCONTROLLER,4MHZ,SO-8	N	N	N	NA	NA	NA	

3130-02616-00	NA	A	IC,NC7WZ17,DUAL SCHMITT TRIGGER BUFFER	N	N	N	NA	NA	NA	

Disposition Codes:
N = New Part/Document, no stock
R = Rework
S = Scrap

U = Use As is
NA = Not Applicable

Voting Approval Process: ECO Approval, E1068-0-1
Started: 09-APR-2003 (1 Days)

Done	Who	Activity	Started	Duration	Result
✓	Lindsay, Joshua	Vote Approve/Reject	09-APR-2003 15:23:41	17 Hours 27 Minutes	Approve
✓	Schmidt, Nancy	Vote Approve/Reject	09-APR-2003 15:23:41	17 Hours 27 Minutes	Approve
✓	Elliott, Mr. Gregory	Vote Approve/Reject	09-APR-2003 15:23:41	17 Hours 27 Minutes	Approve
✓	Burkhart, Brad	Vote Approve/Reject	09-APR-2003 15:23:41	17 Hours 27 Minutes	Approve

View Diagram Advanced Options

```
 Volume in drive C has no label.
 Volume Serial Number is 8C84-5F3A

 Directory of C:\source\MuxCode\muxrev0

09/30/2005  02:16 PM    <DIR>          .
09/30/2005  02:16 PM    <DIR>          ..
05/09/2003  11:31 AM               354 avrBuild.bat
05/09/2003  11:29 AM             1,972 init.asm
05/01/2003  04:15 PM             1,084 main.asm
05/21/2003  10:50 AM               695 monmux.hpp
05/08/2003  12:00 PM             1,721 mux.asm
05/19/2003  03:04 PM             2,190 mux.hpp
05/09/2003  11:28 AM             1,932 mux.inc
07/14/2003  04:47 PM             3,649 muxRev0.aps
05/09/2003  11:31 AM               693 muxrev0.hex
05/09/2003  11:31 AM            17,404 muxrev0.lst
05/09/2003  11:31 AM             3,925 muxrev0.map
05/09/2003  11:31 AM             1,370 muxrev0.obj
05/08/2003  01:50 PM             2,893 muxrevt.hex
05/01/2003  04:21 PM             1,152 Regdefs.asm
11/06/2001  03:18 PM             3,833 tn12def.bak
05/01/2003  04:27 PM             4,384 tn12def.inc
05/09/2003  11:30 AM             3,080 uartisr.asm
05/09/2003  11:28 AM             1,490 Vectors.asm
              18 File(s)         53,821 bytes
               2 Dir(s)  16,103,907,328 bytes free
```

EXHIBIT E

```
AVRASM ver. 1.57  C:\source\MuxCode\muxRev0\main.asm Fri May 09 11:31:37 2003

C:\source\MuxCode\muxRev0\main.asm(28): warning: A .db segment with an odd number of bytes is detected. A zero byte is added.
          ;-----------------------------------------------------------------------------------------------
          ;      Mux Code - main.asm
          ;
          ;      Pillar Data Systems - Copyright (C) 2003
          ;-----------------------------------------------------------------------------------------------
          ;

          ;-----------------------------------------------------------------------------------------------
          ;
          ; Main is just a collection of includes since there is no linker. Everything is assembled
          ; in order as one big file.
          ;
          ;-----------------------------------------------------------------------------------------------

           .nolist
           .include      "mux.inc"                                          ;Hardware header file
          ;-----------------------------------------------------------------------------------------------
          ;      Mux Code - mux.inc
          ;
          ;      Pillar Data Systems - Copyright (C) 2003
          ;-----------------------------------------------------------------------------------------------
          ;

          ;-----------------------------------------------------------------------------------------------
          ;
          ; This file contains the design unique register and bit definitions
          ;
          ;-----------------------------------------------------------------------------------------------
          ;
          ;system parameters
          ;
           .equ  cpu_clk         =       1200000                    ; 1.2MHz clock

          ;portb bit definitions

           .equ  waggle          =       0                                ; wiggle this pin to show that UART clock is running
           .equ  pwr_control     =       1                                ; Drive power control
           .equ  mux_control     =       2                                ; Mux control
           .equ  rc1_rxd         =       3                                ; RAID Controller 1 receive data
           .equ  rc0_rxd         =       4                                ; RAID COntroller 0 receive data
           .equ  rst                     =       5                                ; Not used as I/O, only as power on reset

           .equ  sw_baud         =       100

          ;timer parameters

           .equ  time            =       (256 - 48)                 ; Baud rate = clk/(8 * 50 * 30) = 100 (tuned slightly)

          ;control characters

           .equ  power_on        =       0x05                       ; Drive power on command
           .equ  power_off       =       0x0a                       ; Drive power off command
                                                                 Page 1
```

EXHIBIT F

```
.equ  mux_sel0        =       0x09                    ; Select mux port 0
.equ  mux_sel1        =       0x0d                    ; Select mux port 1

;
; UART status register bit definitions
;
.equ  tx1_empty       =       7                       ; Transmit buffer empty flag, uart 1
.equ  tx2_empty       =       6                       ; Transmit buffer empty flag, uart 2
.equ  rx1_full        =       5                       ; Receive buffer full flag, uart 1
.equ  rx2_full        =       4                       ; Receive buffer full flag, uart 2
.equ  rx1_over        =       3                       ; Receive buffer overrun, uart 1
.equ  rx2_over        =       2                       ; Receive buffer overrun, uart 2

;
;system configuration
;
.equ  bitRingCnt              =       30              ; 30 clocks per UART bit
.equ  StartRingCount  =       45              ; 45 clocks from start transition to center of first bit cell

;
; End of mux.inc
;

.include      "regdefs.asm"                           ;Register definitions
;-----------------------------------------------------------------------------------------------
;     Mux Code - regdefs.asm
;
;     Pillar Data Systems - Copyright (C) 2003
;-----------------------------------------------------------------------------------------------
;

;-----------------------------------------------------------------------------------------------
;
; Named processor register definitions
;
;-----------------------------------------------------------------------------------------------
;
; r0 is used for the LPM instruction and is general purpose
;

;
; UART.asm Register definitions. The s/w uarts are very register intensive
;
.def  sw_uart_stat    =       r18                     ;Status register
.def  rx1_sreg                =       r19             ;Receive shift reg, uart 1
.def  rx2_sreg                =       r20             ;Receive shift reg, uart 2
.def  rx1_bring               =       r21             ;Receive bit ring, uart 1
.def  rx2_bring               =       r22             ;Receive bit ring, uart 2
.def  rx1_state               =       r23             ;Receive state, uart 1
.def  rx2_state               =       r24             ;Receive state, uart 2
.def  rx1_data                =       r25             ;Receive data, uart 1
.def  rx2_data                =       r26             ;Receive data, uart 2

;
; end of regdefs.asm
;         .include    "vectors.asm"                   ;Reset & Interrupt Vectors
;-----------------------------------------------------------------------------------------------
```

```
;       Mux Code - vectors.asm
;
;       Pillar Data Systems - Copyright (C) 2003
;-------------------------------------------------------------------------------
;

;-------------------------------------------------------------------------------
;
; This file contains all of the reset and interrupt vectors and must be the first .asm
; file to be assembled.
;
;-------------------------------------------------------------------------------

        .cseg
        .org    0x0                                             ;Vectors are at start of memory

;Interrupt vectors

000000 c007             rjmp    reset                           ; Reset Handler
000001 c005             rjmp    spurious                        ; IRQ0 Handler
000002 c004             rjmp    spurious                        ; Pin Change Handler
000003 c01d             rjmp    timer_tick                      ; Timer0 Overflow Handler
000004 c002             rjmp    spurious                        ; EEPROM Ready Handler
000005 c001             rjmp    spurious                        ; Analog Comparator Handler

;
; Process spurious interrupt
;
osc_calibration:
        .db     0xff,0xff                                       ; reserve space for oscillator calibration byte
000006 ffff

spurious:
000007 9518             reti                                            ; Return from spurious interrupt

;
; Reset vector code. This copies the oscillator calibration byte into the calibration register.
; This then falls through to the init.asm module.
;
reset:
000008 e0ec             ldi             r30,low(osc_calibration * 2)    ; Point to calibration byte
000009 27ff             clr             r31
00000a 95c8             lpm                                                     ; get byte from flash
00000b be01             out             OSCCAL,r0                               ; put it in the calibration reg

;
; End of vectors.asm
;
.include        "init.asm"                                      ;Hardware Initialization
;-------------------------------------------------------------------------------
;       Mux Code - init.asm
;
;       Pillar Data Systems - Copyright (C) 2003
;-------------------------------------------------------------------------------
;

;-------------------------------------------------------------------------------
```

```
                        ;
                        ; Initialize the various hardware registers & peripherals
                        ;
                        ;-----------------------------------------------------------------------------------------
                        init:
                        ;
                        ; initialize registers
                        ;

                        ;
                        ; Initialize portb
                        ;
                        ; Port b is mixed I/O. Bits 3 & 4 are inputs with pull-ups, bits 0,1 & 2 are
                        ; outputs.
                        ;

00000c e007             ldi     r16,(1<<pwr_control)+(1<<mux_control)+(1<<waggle)       ; Make 3&4 inputs, 0,1&2 outputs
00000d bb07             out     ddrb,r16

00000e e108             ldi     r16,(1<<rc1_rxd)+(1<<rc0_rxd)           ; Set up pullups on bits 3&4
00000f bb08             out     portb,r16                                               ; This will also hold drive power off and mux
control = low

                        ;
                        ; Initialize soft UARTs
                        ;

                        init_sw_uart:
000010 2722             clr             sw_uart_stat                                    ; Uart status register
000011 2733             clr             rx1_sreg                                        ; Receive shift reg, uart 1
000012 2744             clr             rx2_sreg                                        ; Receive shift reg, uart 2
000013 e15e             ldi             rx1_bring,bitRingCnt                    ; Receive bit ring, uart 1
000014 e16e             ldi             rx2_bring,bitRingCnt                    ; Receive bit ring, uart 2
000015 2777             clr             rx1_state                                       ; Receive state, uart 1
000016 2788             clr             rx2_state                                       ; Receive state, uart 2
000017 2799             clr             rx1_data                                        ; Receive data, uart 1
000018 27aa             clr             rx2_data                                        ; Receive data, uart 2

                        ;
                        ; Initialize Timer
                        ;
                        ; We will initialize timer 0 to provide a periodic interrupt
                        ; which is used to clock the software UARTs.
                        ;
000019 ed00             ldi             r16,time
00001a bf02             out             TCNT0,r16                                       ; Set up timer to wrap such that
                                                                                        ; it interrupts at 25 x the sw uart
00001b e002             ldi             r16,2                                           ; baud rate
00001c bf03             out             TCCR0,r16                                       ; prescale divide by 8
00001d e002             ldi             r16,(1<<TOIE0)
00001e bf09             out             TIMSK,r16                                       ; enable timer interrupt

                        ;
                        ; Go do main loop
                        ;
00001f 9478             sei                                                             ; enable processor interrupts
```

```
000020 c035            rjmp    main_loop

        ;
        ; End of init.asm
        ;
        .include      "uartisr.asm"                                ;Soft uart code & timer isr
        ;-----------------------------------------------------------------------------------------------
        ;       Mux Code - uartisr.asm
        ;
        ;       Pillar Data Systems - Copyright (C) 2003
        ;-----------------------------------------------------------------------------------------------
        ;

        ;-----------------------------------------------------------------------------------------------
        ;
        ; Initialize software UART registers
        ;
        ;-----------------------------------------------------------------------------------------------


        ;-----------------------------------------------------------------------------------------------
        ;
        ; Process timer interrupt and UARTs
        ;
        ;-----------------------------------------------------------------------------------------------

        timer_tick:

000021 b71f            in              r17,SREG                                ; save status register
000022 ede0            ldi             r30,time                                ; reload timer with time value
000023 bfe2            out             TCNT0,r30

        ;
        ; Process UART stuff
        ;

        ;
        ; uart 1
        ;
        uart1_0:
000024 9ac0            sbi             PORTB,waggle                            ; set pin to see if running
000025 955a            dec             rx1_bring
000026 f419            brne    uart1_1                                         ;
000027 e15e            ldi             rx1_bring,bitRingCnt            ; wrap bit ring
000028 3070            cpi             rx1_state,0                                     ; see if state 0
000029 f441            brne    uart1_2                                 ; go see if next state

        uart1_1:
00002a 3070            cpi             rx1_state,0                                     ; see if state 0
00002b f481            brne    uart2_0                                 ; go process other uart
00002c 99b4            sbic    PINB,rc0_rxd                    ; see if start bit
00002d c00e            rjmp    uart2_0

00002e e25d            ldi             rx1_bring,StartRingCount        ; wait until mid bit cell of first data bit
00002f 9573            inc             rx1_state
000030 2733            clr             rx1_sreg
```

```
000031 c00a             rjmp    uart2_0                         ; go process other uart

        uart1_2:
000032 3076             cpi             rx1_state,6                             ; see if state 1
000033 f029             breq    uart1_3                         ; go see if next state

000034 9535             asr             rx1_sreg
000035 99b4             sbic    PINB,rc0_rxd                    ; see if bit is a 1
000036 6130             ori             rx1_sreg,0x10                   ; put in future lsb
000037 9573             inc             rx1_state
000038 c003             rjmp    uart2_0                         ; go process other uart

        uart1_3:
000039 2f93             mov             rx1_data,rx1_sreg               ; transfer data from shift register to data register
00003a 6220             sbr             sw_uart_stat,(1<<rx1_full)      ; set uart status to "rx_full"
00003b 2777             clr             rx1_state                       ; reset state machine

        ;
        ; uart 2
        ;
        uart2_0:
00003c 956a             dec             rx2_bring
00003d f419             brne    uart2_1                         ;
00003e e16e             ldi             rx2_bring,bitRingCnt            ; wrap bit ring
00003f 3080             cpi             rx2_state,0                             ; see if state 0
000040 f441             brne    uart2_2                         ; go see if next state

        uart2_1:
000041 3080             cpi             rx2_state,0                             ; see if state 0
000042 f481             brne    end_isr
000043 99b3             sbic    PINB,rc1_rxd                    ; see if start bit
000044 c00e             rjmp    end_isr

000045 e26d             ldi             rx2_bring,StartRingCount        ; wait until mid bit cell of first data bit
000046 9583             inc             rx2_state
000047 2744             clr             rx2_sreg
000048 c00a             rjmp    end_isr                         ; go finish isr

        uart2_2:
000049 3086             cpi             rx2_state,6                             ; see if state 1
00004a f029             breq    uart2_3                         ; go see if next state

00004b 9545             asr             rx2_sreg
00004c 99b3             sbic    PINB,rc1_rxd                    ; see if bit is a 1
00004d 6140             ori             rx2_sreg,0x10                   ; put in future lsb
00004e 9583             inc             rx2_state
00004f c003             rjmp    end_isr                         ; go finish isr

        uart2_3:
000050 2fa4             mov             rx2_data,rx2_sreg               ; transfer data from shift register to data register
000051 6120             sbr             sw_uart_stat,(1<<rx2_full)      ; set uart status to "rx_full"
000052 2788             clr             rx2_state                       ; reset state machine

        ;
        ; Finish up ISR
        ;
        end_isr:
```

```
000053 98c0            cbi         PORTB,waggle                ; clear pin to see if running
000054 bf1f            out         SREG,r17                            ; restore status register
000055 9518            reti

        ;
        ; end of uartisr.asm
        ;       .include    "mux.asm"                           ;Actual command processor/main code
        ;-----------------------------------------------------------------------------------------
        ;       Mux Code - mux.asm
        ;
        ;       Pillar Data Systems - Copyright (C) 2003
        ;-----------------------------------------------------------------------------------------
        ;


        ;-----------------------------------------------------------------------------------------
        ;
        ; This code just polls the two UARTs and executes the commands it may receive.
        ;
        ;-----------------------------------------------------------------------------------------

        main_loop:

000056 fd25            sbrc    sw_uart_stat,rx1_full
000057 c003            rjmp    process_command0
000058 fd24            sbrc    sw_uart_stat,rx2_full
000059 c005            rjmp    process_command1
00005a cffb            rjmp    main_loop

        process_command0:
00005b 7d2f            cbr             sw_uart_stat,(1<<rx1_full)
00005c 2f09            mov             r16,rx1_data                ;get command from uart
00005d 700f            andi    r16,0x0f                            ;mask out command bits only
00005e c003            rjmp    process_command

        process_command1:
00005f 7e2f            cbr             sw_uart_stat,(1<<rx2_full)
000060 2f0a            mov             r16,rx2_data                ;get command from uart
000061 700f            andi    r16,0x0f                            ;mask out command bits only

        process_command:
000062 3005            cpi             r16,power_on                ;see if power on command
000063 f411            brne    process_command10

000064 9ac1            sbi             PORTB,pwr_control           ;turn on power
000065 cff0            rjmp    main_loop

        process_command10:
000066 300a            cpi             r16,power_off               ;see if power off command
000067 f411            brne    process_command20

000068 98c1            cbi             PORTB,pwr_control           ;turn off power
000069 cfec            rjmp    main_loop

        process_command20:
00006a 3009            cpi             r16,mux_sel0                ;see if SATA port 0 select
00006b f411            brne    process_command30
```

```
00006c 98c2             cbi             PORTB,mux_control                       ;switch mux
00006d cfe8             rjmp    main_loop

        process_command30:
00006e 300d             cpi             r16,mux_sel1                            ;see if SATA port 1 select
00006f f731             brne    main_loop                                       ;invalid command, wait for another

000070 9ac2             sbi             PORTB,mux_control                       ;switch mux
000071 cfe4             rjmp    main_loop

        ;
        ;
        ; end of mux.asm
        ;
        ;
        ; Code version tag
        ;
        .db   "0.0.0"                                                           ;Revision string
000072 2e30
000073 2e30
000074 0030

        ;
        ; end of main.asm
        ;
Assembly complete with no errors.
```